Kim-1/6502 USER NOTES

SEPTEMBER 1976 REPRINT VOLUME 1 ISSUE 1

First of all, I would like to thank you for your response to the newsletter. There are about 200 of us now and our number continues to grow daily. The creativity in this group has not ceased to amaze me since the first letters started rolling in. The most noteworthy has been in the area of games, and demonstration programs in general without the use of any additional I/O devices.

Instead of standing there with a blank look on your face when your family and/or friends say "That's nice, but what can it do?", you can load up with a neat demo program and dazzle them to no end. Jim Butterfields' Real-time Lunar Lander should go a long way toward this end (and you don't need a terminal).

It would be beneficial to publish a list of all the members in the next issue so local people can communicate directly. If you don't want yourname & address published for any reason, please drop me a note very soon and I'll keep your name off the list.

Several Amateur Radio operators have suggested starting a Kim Users Net to meet on the ham bands. This is an excellent idea - as there are quite a few Hams in the group (myself included). Would anyone like to coordinate activities in this area?

••••••••••••••••••••

Any of you wishing to start local groups of Kim-1/6502 users should let me know - I'll pass it on. Here's one:

Buffalo New York - Kim-1 group no forming.
Call (716) 634-6844 for information.

••••••••••••••••••••

Have you been waiting for an assembler, or perhaps Tiny Basic? Well wait no longer.

Kim-1 Tiny Basic for $5.00 is now being offered by: Itty Bitty Computers
P.O. Box 23189
San Jose, Calif. 95153

Con't.

Included in the deal, is a 26 page user manual, a hexidecimal paper tape and a tape in the MOS Technology loader format. A hex listing will be substituted for the tape for those who don't have paper tape equipment. They also have versions of TB for 6800 systems, Jolt, Apple, and Homebrew 6502 systems.

.....

A note from: Bob Grater, Microfit Systems, 1595-21 Laurelwood Rd., Santa Clara, Calif. 95050 Phone (408) 246-4813

Just a short one with some infor for the next newsletter. Steve Pittman here in San Jose has just finished a 2K "Tiny Basic" for KIM, and we'll have it on the market probably in the next couple of weeks (will give you more info as things jell). It should run about $7.00 on paper tape ppd. and will be sold thru the "Byte Shop" in Santa Clara.

What we would like to do is get it on cassettes and offer it in that form for the basic price plus the cost of a cassette. I don't have any additional RAM tacked onto KIM yet and would like to find someone in the area that has at least 2K of memory so we can transfer the BASIC on to a master cassette. A free copy on either paper tape or cassette will go to anyone who will help us on this and they may contact me at the above address & number.

Will let you know when everything is tied-down as to form, price, ect.

I completed my SAB-1 interface adaptor board and we are having the boards made now. The complete kit should be available by 1 Oct. from the "Byte Shop #2" in Santa Clara. Don't have price yet and am researching that now (estimate around $25). The adaptor board will interface KIM's TTY port to any parallel TV Typewriter and Keyboard, with selectable baud rates from 100 to 1760. It is aimed at an easy jumper selectable interface with the "Byte Shop's" TVT kit that runs $120 for the kit and $35 for the bare board. This should give a good alternative to anyone who doesn't want to go out and spend 900 clams for a TTY! (and they are noisey).

Still don't have Tom Pittman's 2K TINY BASIC in hand, but it should be here soon and will let you know on that also.

Just to give you an update on the latest happenings "out this Way".

CU

Bob Grater

.....

Editors Note:

I've heard mention of a method for improving KIM TTY operation by raising the voltage on the TTY resistive divider network R48,R49. Can anyone supply the data on this modification?

OUR FILE OF ARTICLES FOR UPCOMING ISSUES IS RUNNING LOW!!! PLEASE SEND MORE MATERIAL!!! ANY SUBJECT THAT INTERESTS YOU WILL INTEREST OTHER MEMBERS!!!!

.....

To prevent duplication of efforts in the areas of software and hardware design projects for the Kim or 6502 based machine, the User Notes will publish your name together with others who are working on similar projects.

Need some ideas?

Here's a note from Jim Butterfield, 14 Brooklyn Avenue, Toronto M4M 2X5 Canada.

Thought I'd produce something to fire up the imaginations of those KIMmers who'd like to start a project, but just can't think of a good one. The dozen I outline are just a sample... there are lots more, like GOLF, FACTORS OF AN INTEGER, MAD CHEMIST, etc. etc... this list could go on endlessly; many are games, some are physically useful. I've stayed out of diagnostics (SCAN MEMORY, RELOCATE PROGRAM, etc.), mostly because they could change in character for those lucky enough to have printer/keyboard. And I've tried to make the descriptions fuzzy enough so that the programmer can create a system which is still truly his own concept. Last thought: I want to deliver the idea that you can have lots of fun without a terminal.

Has anyone thought of building a light pen for use with the KIM-1 display?

The following programs are concepts. None of them have been written (by me, at any rate). So if you want to take them on as a project, you'll be creating a new KIM capability. (Don't forget to send 'em to USER NOTES when they're ready; or, if you get stuck, call for help in USER NOTES!).

They can also be used as 'kicking off' points for other concepts; after all, ideas often generate new ideas.

Some projects are tougher than others. You may find one that's exactly your speed.

1. SUPER-CLOCK.

The six digits of the KIM-1 display are ideal for showing time in hours, minutes, and seconds. With the speed of the 6502, your main job will be killing time (no pun intended) until the next second needs to be added.

A simple wait loop will do the job for a basic clock. Each time you call the display (using SCANDS) you'll use up a bit over 3 milliseconds. (Has anyone worked out the exact timing of this subroutine?) So calling the display two or three hundred times before adding to the seconds should get you into the right ball park.

A more advanced approach is to use the KIM-1 timer. This will free you from having to count your loops and instructions so exactly, and you'll be able to add extra goodies to the clock with more freedom.

How about these for super-clock options? (a) 12-hour or 24-hour timing; (b) a Minute-Minder option with audible alarm; (c) stopwatch capability; (d) alarm clock with audible alarm; (e) Westminster chimes or Cuckoo selectable by pushbutton; (f) a chess clock - check with a chess maniac friend for the rules on this one. There are lots more - use your imagination.

Con't.

2. PONY RACE.

Three horses race across the KIM-1 display. They are represented by dashes in the top, middle, and bottom positions respectively. The race in run in, say, four laps; as the first horse moves off the display to the right, the display "hops" over to the next lap (you won't see the following horses until they reach this area). On the final lap, the finish line is visible as a vertical line. When the first horse reaches it, the display freezes, so that bets can be paid off without dispute.
Sample display:

You'll need a random number generator, of course, so the ponies will travel at varing speeds and the winner will be unpredictable. Random number generators are a whole study in themselves, of course. You might like to wright your own; or the following one, which requires six data locations (from RND to RND+5):

```
CLD
SEC
LDA RND+1
ADC RND+4
ADC RND+5
STA RND
LDX #$04
LOOP LDA RND,X
STA RND +1,X
DEX
BPL LOOP
```

The random number (from 0 to 255) will be available in either the accumulator or location RND. Each time you execute the above program, a new random number will be generated.

Be sure to slow the action down so that the horses can be "seen" to move.

3. DECIMAL-HEXIDECIMAL CONVERSION.

This could be handy to have while you're writing programs! The first four digits of the display might represent a decimal number, while the last two digits its hexadecimal equivalent.

You should be able to enter either decimal or hex, and get the equivalent instantly. (How about <u>signed</u> decimal?)

4. LE MANS.

You've got four-on-th-floor and lots of horsepower. Buttons A to D put you into the various gears (A is low, D is high, and maybe E is neutral).

Con't.

Your gas pedal is the numbered keys: 9 is maximum power. Choose your gears carefully: too high a gear and you'll stall, except in low gear; too low a gear and you'll lose acceleration as your speed increases. To complicate the situation, you'll blow the motor if your RPM exceed a given limit.

This one should come with sound; the noise of the "motor" is vital to the realism of the game.

When no key is being pressed, the motor is "idling" at minimum power. When you're in gear, the motor RPM and the vehicle speed are locked together in the proper ratio for that gear. Each gas pedal key corresponds to a certain motor RPM, but you won't reach that RPM immediately; your speed will creep up (or down) depending on what gear is engaged.

There are lots of different ways you can set up the display. A digital speedometer is the most obvious. As an alternative, how about "telephone poles" (vertical lines on the display) which move from the center of the display to the outside according to your speed?

5. COSMIC RAY DETECTOR.

A great "do nothing" program. It doesn't really detect cosmic rays, but the effect is nice. Little "blips" dash erratically across the display: up and down, left to right, etc. Many of them have an audio output associated with them (bleep! plunk!)

You can use a random number generator, or simply scan memory for your data (which will be random enough).

If you want to wire out to a temperature/skin resistance detector, you could cause the display rate to change if someone touches the sensor. Might make a great "sex appeal" meter.

6. CRAPS (dice game).

Using the random number generator, roll a pair of dice. Seven and eleven win; two, three, or twelve lose; any other number is a "point" and requires extra rolls.

Each roll may be initiated by pushing a button; or the machine may keep rolling until it wins or loses. The player could "bet" certain amounts before any play; and the computer would keep a tally on how he's done so far.

7. AUNTIE GRISELDA'S FORTUNE TELLING MACHINE.

Think of a question that can be answered Yes or No. Push the button - any button - and Auntie Griselda will answer.

She might just say YES or NO - but you could also insert any other messages that might fit on the KIM-1 display, such as:

dunno
huh?
BUGoff
OY

or even the FRENCH OUI or NON.

Since only one random number is needed each time the button is pushed, you might like to try a true "randomizing" program. Keep incrementing your random value with an INC command until the button is pushed. The result will be truly random, since there's no predicting when the questioner will hit it.

8. SIX-DIGIT CALCULATOR.

Since you can't light the decimal point on the KIM-1 display, you can't do fractions; but you can add and subtract, and even multiply and divide with a little effort.

Decimal arithmetic seems easiest for addition or subtraction, but you'll probably find that you'll be better off doing everything in binary, and then converting for display purposes.

"A" seems to be a good key for Add, with B for Subtract, C for Multiply, and D for divide. E might be good for "Divide and show the remainder", which turns out to be very handy for fixed-point work. And you should have lots of Store (DA) and Recall (AD) capability.

Of course, you can't easily compete with a pocket calculator. But you can sharpen up your arithmetic ability on the KIM, and at least show visitors that the capability is there.

9. NIM.

The game of NIM is pretty well documented, in several texts, but here are the brief rules: you have three piles of objects. The number of objects in each pile is a two-digit number, shown on the display. You play against the computer in the following manner: you may take from any pile, 1 to 9 objects, up to the maximum in that pile. Then the computer plays the same way. Objective: the player who picks up the last remaining objects wins.

If you know the theory, you can program the computer to play a perfect game (providing the deck isn't stacked against it).

The player's interface must be two-stop. First, he must say which pile he wants to draw from; then, how many he wants to take. He must be prevented from taking more units from a pile than exist on the pile; or from drawing from an empty pile.

The game should be set up using random numbers, of course.

How about making the computer less-than-perfect... maybe one out of every then moves or so (randomly) it will make a mistake? How about setting the computer's IQ before playing?

10. TIC-TAC-TOE.

Most people don't give this game enough credit. It has some interesting (but not terribly deep) strategies, particularly if you vary the opening move.

Con't

The board positions are represented by "segments" on the display. Your moves are permanently illuminated; the computer's moves flash on and off.

A simple game, where the computer always takes the first move and always plays a relatively fixed strategy, isn't hard to program - once you have solved the logistics of running the display and keeping trach of the moves (which is a fair sized job).

But if you open up the game - allowing the computer to select randomly among several preferred moves - you can be into quite a project. Might lend itself quite well to a team or class project (arranging to split up a job into several tasks is a whole art in itself).

Don't forget that you can also set the computers IQ before a series of games. Or try this: it gets smarter every time it loses, and dumber every time it wins. So eventually it will always be evenly matched against every opponent!

11. RHYTHM BOX.

This is the type of device that comes on electronic organs to add drums, etc. to the music. It's valuable for professional musicians, but can also come in handy for kazoo bands, or even people who just like to hum along.

First, you have to generate the "sounds". You should have five instruments - four are basically different frequencies:

Bass drum - lowest frequency
Conga - about 50% faster than the Bass
Wood block-much faster (several times) the Conga
Clave - about 50% faster than the Wood block

And finally the Snare drum - this, instead of being a frequency, is "white noise" - a random series of bits.

As a first step, you can set these up to sound when an appropriately numbered key is pressed - 1 for the Bass, 2 for the Conga, etc. (Hardware hackers can build the oscillators externally, if they wish - programming pundits will of course generate the sounds directly in software).

Now comes the slick part. You can set up a program to output many rhythms directly - waltz, march, bossa nova, etc. Be sure to make it variable in speed. Another hint: many professional rhythm boxes have a foot pedal to allow the artist to start and stop it at will.

12. DARKROOM TIMER.

Photography hobbyists will go for this one. More than just a minute minder. This device could: measure the light from the enlarger, and set the print exposure time automatically; control both safelight and enlarger lamp; measure negative contrast and suggest the grade of paper (or variable contrast filter); or even, if desired, compensate for developer temperature and/or exhaustion!

The red LED display doesn't seem to affect most black-and-white papers. It's almost as if it were designed for the job!

Con't.

You'll need relays or SCR circuits, of course, to control power to safelights and enlarger. An audible output might be useful if you want to use KIM to signal developing times (and if you want to control your exposure manually).

Question: Has anybody made an AC controller yet by modifying one of those inexpensive Triac lamp dimmers? I would think an optical coupler would replace the variable resistor and give highly desirable isolation.

Regards,

Jim Butterfield

.....

Some more project ideas from the Editor:

An automated I.C. tester - hardware & software project - should test any standard 7400 series chip (such as 7400, 7404, 7474, 7475, 7490, 7493 etc.) - 1 or 2 I/O ports would be necessary along with support logic. Each type of chip to be tested would require a special software routine. The program would go through all possible signal combinations and indicate go/no go on the KIM display. Open collector logic would require special interface considerations. A low insertion socket would be useful.

Digital Voltmeter, frequency meter, thermometer, capacity meter etc etc. (using the KIM-1 display)

By utilizing the correct input conditioning logic for the desired function any of these possibilities can be realized. For the DVM, see the low cost A/D in the last issue which utilizes a single LM311. A schmitt trigger input could be set up to increment some internal counters for a DFM. The thermometer and capacity meter would be additions to the basic DVM and DFM functions. With all the neat transducer chips coming out, all kinds of possibilities are open.

A programmable function generator - by hooking a D/A chip and a current/voltage converter on an output port you could set programs up to give you - positive ramp, negative ramp, triangular, staircase waveforms or any other complex waveforms you desire.

.....

Along these lines - from Bob McCulla, 20333-15 N.E. #28, Seattle, Wash.98155

One of our people has discovered a little jewel from Motorola - The MC3408 8 bit DAC at $4.00 a shot. He lost little time in connecting it to the applications connector of his KIM board and is using it to control the sweep input of his H.P. function generator. He says that by offsetting the op amp at the output of the DAC, he can get both + and (-) voltage swings out of the unit - works great.

Con't.

Another use of the applications connector could be to control the input pins of the MC14410 touch tone encoder. I use one of these encoders to dial numbers and control one of our local Amateur Radio 144MH FM Repeaters and can see a good use of the KIM system here for telephone number storage and auto repeater control from my car. Along these lines, I understand that some Amateur Radio people in our area are using their systems for contact logging, Morse Code generations and the control of frequency synthesizers and frequency readout units.

A vary interesting article appeared in the EDN Magazine for May 5, 1976, "A/D Conversion Systems : let your uP do the Working". It was written by an Engineer from Motorola and even though it was written about the 6800 uP, the information is directly useable for 6502 systems.

Sincerely,
Bob McCulla

••••••••••••••••••••

Further hints on the KIM cassette interface from John P. Oliver, Asst. Professor of Astronomy, University of Florida, Gainesville, Florida 32611.

Be certain to turn off the interval timer interrupt (READ or WRITE to 1706) before using the audio tape dump routine. Otherwise disaster results - that is, you get unreadable tapes. (This will only be necessary if you have run a program that uses the interval timer.)

For those who wonder what Baud rates are possible with the KIM teletype interface, we have one of ours hooked (through an RS232 converter similar to the one in KUN #1) to a Hazeltine CRT terminal. We can work reliable to 1200 Baud and most of the time we can work at 9600 Baud. At 9600 Baud it sometimes takes several RST, RUBOUT cycles before KIM gets the timing right but the results are well worth it.

Sincerely,
John P. Oliver

••••••••••••••••••••

From Jim Butterfield

LIGHTING THE KIM-1 DISPLAY

A. SIX-DIGIT HEXADECIMAL.

The easiest way to display six digits of data is to use the KIM-1 Monitor subroutine SCAND.

Calling JSR SCAND (20 19 1F) will cause the first four digits to show the address stored in POINTH (00FA and 00FB), while the last two digits of the display show the contents of that address.

If you look at the first three lines of subroutine SCAND (lines 1057 to 1059 on page 25 of the listing), you'll see how the program "digs out" the contents of the address given by POINTL/POINTH and stores it in location INH (00F9). It's neat programming, and worth studying if you're not completely familiar with the 6502's indirect addressing operation.

Thus, if you skip these three lines, and call JSR SCANDS (20 1F 1F) you will be displaying, in hexidecimal, the contents of three locations: POINTH, POINTL, and INH. This, of course, takes six digits.

Con't.

LIGHTING THE KIM-1 DISPLAY - Con't.

To recap: SCAND will display four digits of address and two digits on contents. SCANDS will display six digits of data.

Important: in both cases, the display will be illuminated for only a few milliseconds. You must call the subroutine repeatedly in order to obtain a steady display.

B. DRIVING THE BITS OF THE DISPLAY DIRECTLY.

1. Store the value $7F into PADD (1741). This sets the directional registers.

2. To select each digit of the display, you will want to store the following values in location SBD (1742):

Digit 1: $09
Digit 2: $0B
Digit 3: $0D
Digit 4: $0F
Digit 5: $11
Digit 6: $13

Note that this can easily be done in a loop, adding two to the value as you move to the next digit.

3. Now that you have selected a particular digit, light the segments you want by storing a "segment control" byte into location SAD (1740). The segments will be lit by setting the appropriate bit to 1 in SAD according to the following table:

Bit:	7	6	5	4	3	2	1	0
Segment:	..	Center	Upper Left	Lower Left	Bottom	Lower Right	Upper Right	Top

For example, to generate a small letter "t", we would store $78 (center, upper left, lower left, bottom) into SAD.

4. Now that you have picked a digit and lit the appropriate segments, wait a while. Sit in a delay loop for about ½ milisecond before moving on to the next digit.

THE KIM-1 ALPHABET.

Some letter, like M and W, just won't go onto a 7-segment display. Some, like E, are only possible in capitals; others, like T, can only be done in lower case. So here's an alphabet of possibles:

A - $F7	H - $F6	U - $BE	h - $F4	r - $D0	4 - $E6
B - $FF	I - $86	Y - $EE	i - $84	t - $F8	5 - $ED
C - $B9	J - $9E	b - $FC	j - $9E	u - $9C	6.- $FD
D - $BF	L - $B8	c - $D8	l - $86	y - $EE	7 - $87
E - $F9	O - $BF	d - $DE	n - $D4	1 - $86	8 - $FF
F - $F1	P - $F3	f - $F1	o - $DC	2 - $DB	9 - $EF
G - $BD	S - $ED	g - $EF	p - $F3	3 - $CF	0 - $BF
					minus - $C0

From Stan Ockers, RR#4, Box 209, Lockport, Illinois 60441

Alphanumerics on the KIM Display.

For many, one of the first peripherals they consider adding to their KIM-1 is some sort of alpha-numeric device, a mechanical printer or keyboard and video display unit for example. Unfortunately, such devices involve considerable expense and make your KIM slightly less than portable to say the least. The KIM already has a built in display. Isn't there some way to make use of this for alphanumeric output? Is it worth it? Well, with a few concessions for a slightly strange character set, it is possible as I will try to show. Why not try it on your KIM and see if you think it offers any promise. The following program displays six characters, delays for a set time, shifts each character one space to the left and enters new characters from the right giving a scanning billboard effect.

Basic Display Routine.

The core of the alphanumeric method is contained in the basic display routine starting at Ø269. The subroutine follows the same general pattern of Scand, (page 25 of the KIM listing), as a comparison of the two will show. Where Scand jumps to Convd, (1F2F), however this subroutine picks up characters stored in ØØE8 - ØØED. Convd uses the table starting at 1FE7 to get bytes representing the characters to be displayed. We must generate our own bytes and store them in ØØE8 - ØØED. This is done according to the diagram in figure 1 where the numbers represent bit locations. The most significant bit (8Ø) must be equal to one. For example, to generate a "y" the following bit pattern would be required: 111Ø111Ø. The hex byte representing the letter "y" would then be EE. While the whole alphabet can't be generated, with a little imagination a fairly good substitute can be found. Figure 2 lists the set I've been using. If you wish to use the same table Scand uses and put the hex numbers to be displayed in ØØE8 - ØØED, just jump in at 1F4A instead of 1F4E (Ø277). All that is needed now is some way to get our characters into the zero page locations and some way to continually call this routine while we want a display (a delay-display routine).

Key-Entry Routine.

You may wish to slow down or speed up the display or you may wish to back up to catch something you missed. The key-entry routine allows these functions while the display is operating. Pressing a low numbered key will speed the display up, (shorten delay), while a high value key will slow it down. Zero gives a very long delay (25 sec) and if you press another key after zero you will have to wait for it to time out. Pressing "F" will cause the display to move forward while "B" will make it go backward.

Delay-Display Routine.

The delay display routine (Ø228) is set up in tenth of a second multiples. A ØA in Ø229 give 1Ø tenths or a one second delay. For any other delay you can load X with a multiple of Ø.1 seconds and jump in at Ø22A. The key-entry routine enters the key value in ØØ29, (unless it is a "F" or "B").

Scan Routine.

The routine that picks up the characters to be displayed and puts them in ØØE8 - ØØED starts at Ø2ØØ. You must tell the routine where the characters are located, so Y should contain a pointer value high and the accumulator a pointer value low upon entry. You'll notice that there is no way to get out of this routine unless a null character (ØØ) is encountered, (see Ø21Ø).

Driver.

I've covered the preceding in some detail because you may want to play around with using some of the subroutines individually. All you really need to know about though is the driver and character set. a typical driver to tell where the characters are stored is shown beginning at Ø283. It is set up to use page three. It starts at Ø3Ø1 because you should put a ØØ in Ø3ØØ so that if scanning backwards, the message will turn around. Write your own message starting at Ø3Ø1 using the hex equivalent of characters described in figure 2, (or make up your own), one byte per character. End with a ØØ. The program is started at Ø283. That's all there is to it but no provision has been made for page crossing so you are limited to 256 characters.

I would like to thank Roy Hinman who showed me how to use the display.

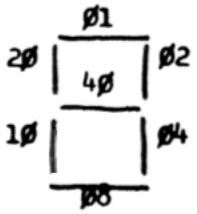


FIG. 1

(Ed. note - only those characters which were not covered on page 10 will be shown here).

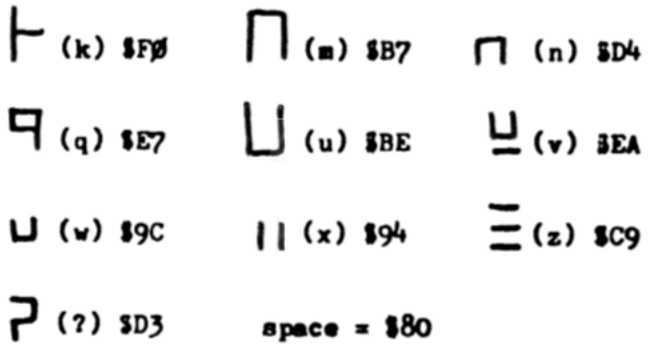


FIG. 2

****SCAN ROUTINE****

```
Ø2ØØ  84  DE      STY ØØDE          ; Transfer pointer high
Ø2Ø2  85  DD      STA ØØDD          ; Transfer pointer low
Ø2Ø4  A9  Ø7      LDA #$Ø7          ; Initialize to scan forward
Ø2Ø6  85  DF      STA ØØDF          ;
Ø2Ø8  AØ  Ø5      LDY #$Ø5          ; Initialize Y
(3)Ø2ØA  A2  Ø5   LDX #$Ø5          ; Initialize X
(2)Ø2ØC  B1  DD   LDA ØØDD (Ind),Y  ; Get character
Ø2ØE  95  E8      STA ØØE8,X        ; Store it
Ø21Ø  C9  ØØ      CMP #$ØØ          ; Last character ?
Ø212  DØ  Ø1      BNE (1)           ; If not, continue
Ø214  6Ø          RTS               ; If sp. exit subroutine
(1)Ø215  88       DEY               ; set up next character
Ø216  CA          DEX               ; set up next store location
Ø217  1Ø  F3      BPL (2)           ; loop if not 6th character
Ø219  D8          CLD               ; Put in binary mode
Ø21A  18          CLC               ; prepare to add
Ø21B  98          TYA               ; get character pointer
Ø21C  65  DF      ADC ØØDF          ; update for new 6 characters
Ø21E  85  DC      STA ØØDC          ; Save new pointer
Ø22Ø  2Ø  28 Ø2   JSR Ø228          ; Jmp to delay-display
```

Con't.

SCAN ROUTINE CON'T.

```
Ø223   A4  DC       LDY ØØDC          ; restore pointer
Ø225   4C  ØA Ø2    JMP (3)           ; continue rest of message

                ****DELAY-DISPLAY ROUTINE****

Ø228   A2  ØA       LDX#$ØA           ; set rate
Ø22A   86  DB       STX ØØDB          ; put in decrement location
(5)Ø22C A9  52       LDA #$52          ; load timer ( 0.1 seconds)
Ø22E   8D  Ø7 17    STA 17Ø7          ; start timer
(4)Ø231 2Ø  3E Ø2    JSR Ø23E          ; jmp to key entry subroutine
Ø234   2C  Ø7 17    BIT 17Ø7          ; timer done?
Ø237   1Ø  F8       BPL (4)           ; if not, loop
Ø239   C6  DB       DEC ØØDB          ; decrement timer
Ø23B   DØ  EF       BNE (5)           ; time completed?
Ø23D   6Ø           RTS               ; go get 6 new characters

                ****KEY ENTRY ROUTINE****

(6)Ø23E 2Ø  8C 1E    JSR INIT 1        ; set up
Ø241   2Ø  69 Ø2    JSR Ø269          ; display-entry
Ø244   dØ  Ø1       BNE (7)           ; skip if key depressed
Ø246   6Ø           RTS               ; exit subroutine
(7)Ø247 2Ø  69 Ø2    JSR Ø269          ; display-entry
Ø24A   DØ  Ø1       BNE (8)           ; key still depressed
Ø24C   6Ø           RTS               ; exit if no
(8)Ø24D 2Ø  6A 1F    JSR GET KEY       ; get key value
Ø25Ø   C9  15       CMP #$15          ; valid key?
Ø252   1Ø  E5       BPL (6)           ; if not - try again
Ø254   C9  ØF       CMP #$ØF          ; forward?
Ø256   DØ  Ø4       BNE (9)           ; if not, continue
Ø258   A9  Ø7       LDA #3Ø7          ; set forward shift
Ø25A   1Ø  Ø6       BPL (10)          ; store it
(9)Ø25C C9  ØB       CMP #$ØB          ; backwards?
Ø25E   DØ  Ø5       BNE (11)          ; if not, continue
Ø26Ø   A9  Ø5       LDA #$Ø5          ; set backward shift
(10)Ø262 85  DF       STA ØØDF          ; store shift
Ø264   6Ø           RTS               ; exit
(11)Ø265 8D  29 Ø2    STA Ø229          ; put key entry in timer
Ø268   6Ø           RTS               ; exit

                ****BASIC DISPLAY ROUTINE****

Ø269   A9  7F       LDA #$7F          ; change seg.
Ø26B   8D  41 17    STA PADD          ; to output
Ø26E   AØ  ØØ       LDY #$ØØ          ; init recall index
Ø27Ø   A2  Ø9       LDX #$Ø9          ; init digit number
(12)Ø272 B9  E8 ØØ    LDA ØØE8,Y        ; get character
Ø275   84  FC       STY ØØFC          ; save Y
Ø277   2Ø  4E 1F    JSR 1F4E          ; display character
Ø27A   C8           INY               ; set up for next character
Ø27B   CØ  Ø6       CMP #$Ø6          ; 6 Char. displayed?
Ø27D   9Ø  F3       BCC (12)          ; if not, get another
Ø27F   2Ø  3D 1F    JSR 1F3D          ; key down?
Ø282   6Ø           RTS               ; exit
```

Con't.

****DRIVER****

```
(13) Ø283   AØ  Ø3      LDY #$Ø3        ; init. point high
     Ø285   A9  Ø1      LDA #$Ø1        ; init. point low
     Ø287   2Ø  ØØ Ø2   JSR Ø2ØØ        ; write
     Ø28A   4C  83 Ø2   JMP (13)        ; repeat
```

DIRECTIONS:

1. Put a ØØ in Ø3ØØ
2. Enter your characters 1 byte per character (see char. set)
3. End with a ØØ
4. Program starts at Ø283
5. A low number pressed speeds u display high number slows it down. A "F" makes it go forward, "B" backward. Ø gives a delay of about 25 seconds between shifts.

****

MOON LANDER by Jim Butterfield - Fly in real time with fuel constraints; this involves interface to display, keyboard monitor; requires active and real-time user interface.

1. Program starts at location 0000. Press AD 0 0 0 0 GO, you will find yourself at 4500 feet and falling. The thrust on your machine is set to low; so you'll pick up speed due to the force of gravity.

2. You can look at your fuel any time by pressing the F button, your fuel (initially 800 pounds) will be shown in the first four digits of the KIM display.

3. You can look at your altitude any time by pressing the A button. Your initial altitude is 4500 feet, and is shown in the first four digits of the KIM display.

4. The last two digits of the KIM display always show your rate of descent or ascent.

5. Set your thrust by pressing buttons 1 through 9. (Warning: button 0 turns your motor off, and it will not reignite! Be prepared for a very hard landing if you press this one!) A thrust of 1, minimum burns very little fuel; but gravity will be pulling your craft down faster and faster. A thrust of 9, maximum, overcomes gravity and reduces your rate of descent very sharply. A thrust of 5 exactly counterbalances gravity; you will continue to descend (or ascend) at a constant rate.

If you run out of fuel, your thrust controls will become inoperative.

6. A safe landing is considered to be one where you land at a descent rate of 5 or less. After you land, your thrust controls will be inoperative, since the motor in automatically turned off; but you can still press F to look at your fuel.

7. Suggestions for a safe flight:

   1. Conserve fuel at the beginning by pressing 1. You will begin to pick up speed downwards.
   2. When your rate of descent gets up to the 90's, you're falling fast enough. Press 5 to steady the rate.
   3. When your altitude reaches about 1500 feet, you'll need to slow down. Press 9 and slow down fast.

Con't.

MOON LANDER - Con't.

4. When your rate of descent has dropped to 15 to 20, steady the craft by pressing 5 or 6. Now you're on your own.

```
0000 A2 0C    GO       LDX #$0C
0002 B5 B8    LP1      LDA INIT, X     'set up initial flite.
0004 95 E2             STA ALT, X
0006 CA                DEX
0007 10 F9             BPL LP1
0009 A2 05    CALC     LDX #$05        acceleration/velocity update
000B A0 01    RECAL    LDY #$01
000D F8                SED
000E 18                CLC
000F B5 E2    DIGIT    LDA ALT, X      add each digit
0011 75 E4             ADC ALT+2,X
0013 95 E2             STA ALT,X
0015 CA                DEX             next digit
0016 88                DEY
0017 10 F6             BPL DIGIT
0019 B5 E5             LDA ALT+3,X
001B 10 02             BPL INCR
001D A9 99             LDA #$99
001F 75 E2    INCR     ADC ALT,X
0021 95 E2             STA ALT,X
0023 CA                DEX
0024 10 E5             BPL RECAL
0026 A5 E2             LDA ALT
0028 10 0B             BPL UP          still flying?
002A A9 00             LDA #$00
002C A2 02             LDA #$02        NOPE, turn off
002E 95 E2    DD       STA ALT,X
0030 95 E8             STA TH2,X
0032 CA                DEX
0033 10 F9             BPL DD
0035 38       UP       SEC             update fuel
0036 A5 ED             LDA FUEL+2
0038 E5 EA             SBC THRUST
003A 85 ED             STA FUEL+2
003C A2 01             LDX #$01
003E B5 EB    LP2      LDA FUEL,X
0040 E9 00             SBC #$00
0042 95 EB             STA FUEL,X
0044 CA                DEX
0045 10 F7             BPL LP2
0047 B0 0C             BCS TANK        Any fuel left?
0049 A9 00             LDA #$00
004B A2 03             LDX #$03        nope, turn off engine
004D 95 EA             STA THRUST,X
004F CA                DEX
0050 10 FB             BPL LP3
              ;show altitude or fuel according to flag
0052 20 AA 00          JSR THRSET
0055 A5 EE    TANK     LDA MODE
0057 D0 0A             BNE SHOFL
0059 A5 E2             LDA ALT
```

Con't.

```
005B A6 E3              LDX ALT+1
005D F0 08              BEQ ST
005F D0 06              BNE ST
0061 F0 A6    LINK      BEQ CALC
0063 A5 EB    SHOFL     LDA FUEL
0065 A6 EC              LDX FUEL+1
0067 85 FB    ST        STA POINTH
0069 86 FA              STX POINTL
              ;show velocity as absolute
006B A5 E5              LDA VEL
006D 30 06              BMI DOWN
006F A5 E6              LDA VEL+1
0071 F0 07              BEQ FLY
0073 D0 05              BNE FLY
0075 38       DOWN      SEC
0076 A9 00              LDA #$00
0078 E5 E6              SEC VEL+1
007A 85 F9    FLY       STA INH
              ;display the bird
007C A9 02              LDA #$02          'suddenness' factor
007E 85 E1              STA DECK
0080 20 1F 1F Flite     JSR SCANDS
0083 F0 06              BEQ NOKEY
0085 20 6A 1F           JSR GETKEY
0088 20 91 00           JSR DOKEY
008B C6 E1    NOKEY     DEC DECK
008D D0 F1              BNE FLITE
008F F0 D0              BEQ LINK
              ;subroutine for reading keys
0091 C9 15    DOKEY     CMP#$15           fuel mode?
0093 D0 03              BNE NALT
0095 85 EE              STA MODE
0097 60                 RTS
0098 C9 10    NALT      CMP #$10          altitude mode?
009A D0 05              BNE NAL2
009C A9 00              LDA #$00
009E 85 EE              STA MODE
00A0 60       RET1      RTS
00A1 10 FD    NAL2      BPL RET1
00A3 AA                 TAX
00A4 A5 EA              LDA THRUST        dead stick?
00A6 F0 F8              BEQ RET1
00A8 86 EA              STX THRUST
00AA A5 EA    THRSET    LDA THRUST
00AC 38                 SEC
00AD E9 05              SBC #$05
00AF 85 E9              STA TH2+1
00B1 A9 00              LDA #$00
00B3 E9 00              SEC #$00
00B5 85 E8              STA TH2
00B7 60                 RTS
00B8          ;INIT     45/00/00//99/80/00//99/98//02//08/00/00//00
to 00C5                 (height) (speed)   (acc) (thr) (fuel)   (mode)
```

NOTE: Moon Lander sometimes slips into the fuel mode by mistake... the cure? Simply change $00B2 to $14 and use the E key for energy (fuel) instead of F.

.....

FROM THE EDITOR:

I would like to clear up any misunderstanding there may be concerning the relationship between our User Notes and MOS Technology. This newsletter is being financed wholly by the subscribers. MOS was kind enough to print and mail the complementary issue so the newsletter could get started but we are self supporting now.

If you didn't get a particular application note or if you want your address changed on MOS Technology mailing list - please write to - KIM-1 Customer Support
c/o MOS Technology
950 Rittenhouse Rd.
Norristown, Pa. 19401

Any correspondence concerning the User Notes should, of course, be addressed to: KIM-1 User Notes
c/o Eric C. Rehnke
7656 Broadview Rd. #207
Parma, Ohio 44134

The subscription rate is $5.00 for issues #1 thru #6 which includes 1st Class postage for U.S. and Canadian subscribers. Foreign subscribers should write for rates.

Payments should be made with check or money order in U.S. funds, no cash or purchase orders please.

FOR YOUR INFORMATION:

Contact for course hardware etc., from Microprocessor Seminar: Joe Williams, School of Engineering, Materials Engineering Dept., Rensselaer Polytechnic Institute, Troy, New York 12181.

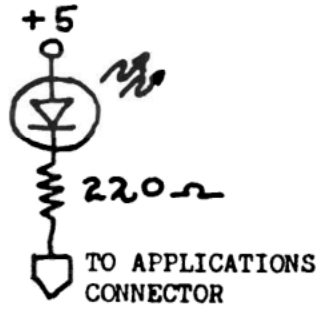
Machine Language Programming for the 8008 and similiar microcomputers $19.95: Scelbi Computer Consulting, Inc., 1322 Rear Boston Post Road, Dept. DJ, Milford Connecticut 06460.

.....

From Robert G. Lloyd, 7554 Southgate Rd., Fayetteville, N. C. 28304:

Here is my program for THERE'S MORE TO BLINKING LIGHTS THAN MEETS THE EYE (January 76, BYTE, Page 52)

```
0300 A9 FF    Caterpil LDA #$FF
0302 8D 01 17          STA PADD Set PADD & PBDD to
0305 8D 03 17          STA PBDD OUTPUT
0308 38                SEC
0309 A9 0F    Newmove  LDA #$0F
030B 8D 00 17          STA PAD  Set march on I/O
030E 8D 02 17          STA PBD  output
0311 A2 FF    Loop 1   LDX #$FF
0313 A0 55    Loop 2   LDY #$55 Set speed of bug
0315 88                DEY
0316 D0 FD             BNE LOOP 2
0318 CA                DEX
0319 D0 F8             BNE LOOP 1
031B 2E 0A 03          ROL
031E 4C 09 03          JMP NEWMOVE
```



INSTALL ONE LED/RESISTOR ON EACH OF THE FOLLOWING APP. CONNECTOR PINS IN THE FOLLOWING ORDER: 14,9,4,10,3,11,2,12,5,13,6,16, 7,15,8. (the first LED goes to pin 14, the second goes to 9, etc.)

.....

From H. T. Gordon, Univ. of Calif., College of Agriculture, Berkeley, Calif. 94720

Listing of Third Version of HEDEC. Converts 4-digit hex number in 00 E6 (hi byte) and 00 E7 (lo byte) into decimal equivalent stored in 00 E0, 00 E1, and 00 E2. Uses 00 E3, 00 E4, and 00 E5 to store calculated conversion factors for each of 16 binary bits. Length: 67 bytes. Conversion times: 0.7 millisec for hex 0000, 1.5 for hex 1111, 1.4 for hex 8080, 2.12 for hex FFFF. Times are proportional to the number of binary 1 bits, not to the numerical value.

```
0200 F8          (sets decimal mode)
     98          (pushes Y, then X, index into stack)
     48
     8A
     48
0205 A9 00       (zeroes 00 E0 to 00 E5 in a loop)
     A2 06       (sets X-index for 6 operations)
     95 DF       (zero-page, X storing)
     CA
020C D0 FB
     E6 E5       (increments 00 E5 to 01, to be first conversion factor)
0210 A5 E7       (accumulator pick-up of lo hex byte)
0212 48          (stored in stack)
     A0 08       (sets Y index for testing of 8 bits)
0215 68          (pulls hex byte from stack)
     4A          (one logical shift right, lowest bit in carry)
     48          (stores shifted hex byte in stack)
0218 90 0C       (if carry clear, bit was a zero, skip to 0226)
     A2 03       (if not, do triple-precision add of conversion factor to the decimal
     18           locations)
021D B5 E2
     75 DF
     95 DF
     CA
0224 D0 F7
0226 A2 03       (next conversion factor always calculated, doubling previous factor
     18           by adding it to itself, giving sequence 1, 2, 4, 8,..... to final
     B5 E2        65536 (not used))
     75 E2
     95 E2
     CA
0230 D0 F7
     88          (DEY)
0233 D0 E0       (if not zero, back to 0215 for next bit)
0235 68          (this PLA stack pull needed to equalize PHAs and PLAs)
     A5 E3       (LDA highest conversion factor location)
0238 D0 04       (if not zero, job is finished, so exit)
     A5 E6       (if zero, load hi hex byte)
023C D0 D4       (if not zero, back to 0212 for bit testing)
023E 68          (restore X, then Y, indexes)
     AA
     68
     A8
0242 D8          (clear decimal mode)
0243 60          (RTS)
```

.....